Behaviors of NIST Utilities on Items Where the I++ DME Spec is not Explicit

Thomas Kramer
The National Institute of Standards and Technology (NIST)
Telephone: 301.975.3518
Email: thomas.kramer@nist.gov

February 4, 2004

Introduction

This write-up describes how the NIST utilities behave in cases where the I++ DME Interface specification is not explicit about how they should behave.

In this write-up "utilities" means version 2.1 of the NIST I++ DME test suite, and "spec" means release 1.30 of the I++ DME Interface specification. Where release 1.4 contains additional information, this is mentioned and ascribed explicitly to release 1.4. Page numbers and section references in this write-up refer to specific sections in the release 1.30 of the I++ DME Interface specification.

Topics are arranged alphabetically by title.

Comments within each topic consist of three sections. The "A." section describes what the spec says. The "B." section says how the utilities work. The "C." section says how the utilities correspond to the spec.

1 AlignPart data response

A. The spec is not clear about how many vectors should be included in a data response message for an AlignPart command. The phrase "the same number as set" seems to imply that if the command has two vectors in it, the response should have one vector, and if the command has four vectors in it, the response should have two vectors. (Sec. 12.1).

B. The server utility puts one vector in the response if the command has two vectors. The utility puts two vectors in the response if the command has four vectors.

C. We guess that this is what the spec intends, but we are unsure.

2 AlignTool data response

A. The data in a response to AlignTool is supposed to be "vectors". When there are two vectors, it is not clear how the data is supposed to be returned. There are at least three reasonable ways to return data:

   1. Put (i1, ji, ki, i2, j2, k2) in a single response.
   2. Put ((i1, ji, ki), (i2, j2, k2)) in a single response.
   3. Send two responses with one vector in each.

   The spec does not specify which, if any, of these is intended. (Sec. 6.3.2.20).

B. The sender utility puts (i1, ji, ki, i2, j2, k2) in a single response.

C. We do not know if what the utilities do is what the spec intends.

3 ChangeTool command arguments

A. It is not clear in the spec whether it is allowed to use "UnDefTool" "BaseTool", "RefTool", or "NoTool" as an argument to ChangeTool. (Sec. 6.3.2.18).

B. The utilities allow "BaseTool", "RefTool", and "NoTool" as arguments to ChangeTool, but not "UnDefTool".

C. We do not know if what the utilities do is what is intended by the spec.

4 EnumProp and EnumAllProp command arguments

A. The spec is not clear about what arguments are allowed for EnumProp and EnumAllProp. (Sec. 6.3.1.12 and Sec. 6.3.1.13).

   In addition to the general lack of clarity, the spec does not disallow asking for properties of UnDefTool when UnDefTool is FoundTool. However, UnDefTool is conceptually undefined, so it seems implicit that it should not have properties. (Sec. 6.3.2.14).

B. The rules in the utilities are:

   1. The argument must have one of the following patterns.
      - key1()
      - key1.key2()

- key1.key2.key3()

2. key1 must be FoundTool or Tool.
3. key2 must be GoToPar or PtMeasPar.
4. If key2 is GoToPar, key3 must be one of: Speed, MinSpeed, MaxSpeed, Accel, MinAccel, MaxAccel.
5. If key2 is PtMeasPar, key3 must be one of: Speed, MinSpeed, MaxSpeed, Accel, MinAccel, MaxAccel, Approach, MinApproach, MaxApproach, Retract, MinRetract, MaxRetract, Search, MinSearch, MaxSearch.

The context checker (and, hence, the server utility), returns an error if an EnumProp or EnumAllProp command asks for properties of FoundTool when FoundTool is UnDefTool.

C. We do not know if what the utilities do is what is intended by the spec.

5 EnumProp and EnumAllProp data responses

A. In the spec, the descriptions of EnumProp and EnumAllProp are unclear. The difference between the the two is also unclear. It is not possible to tell from the text of the spec what responses should be returned by these two commands. (Sec. 6.3.1.12 and Sec. 6.3.1.13).

In release 1.4, examples 7.7 (EnumProp) and 7.8 (EnumAllProp) add more confusion. Example 7.7 contains multiple inconsistent responses, while some responses in Example 7.8 are identical to responses in Example 7.7.

B. In the server utility data responses, EnumProp returns the name of the type of a property, while EnumAllProp returns the names and types of the immediate children of a property. Under this interpretation, if the command, for example, is EnumProp(Tool.GoToPar()), there is one response message with data as follows:

- "GoToPar", "Property"

If the command is EnumAllProp(Tool.GoToPar()), there are six response messages, with data as follows:

- "MinSpeed", "Number"
- "Speed", "Number"
- "MaxSpeed", "Number"

- "MinAccel", "Number"
- "Accel", "Number"
- "MaxAccel", "Number"

C. We suspect that what the utilities do is not what is intended by the spec, but we cannot tell what the spec intends.

6 EnumTools data response

A. The spec does not say specifically which tools should be included in the response to EnumTools. It seems implicit that all the predefined tools (BaseTool, RefTool, NoTool and UnDefTool) should be included. (Sec. 6.3.2.23). However, in release 1.4 (Sec. 6.3.2.14), it says explicitly that BaseTool and UnDefTool should not be included.

B. The server utility lists all four predefined tools in the data responses to EnumTools.

C. What the utilities do apparently differs from what is intended by the spec. This will be changed in the release 3.0 of the utilities.

7 Error codes

A. The spec defines 47 command error messages (Sec. 8.2), and the descriptions of the commands list some of the error messages that can result from each command. These listings, however, are grossly incomplete and do not make sense in some cases. The spec does not deal with response errors.

B. The utilities have 157 error messages for command parsing, 52 error messages for response parsing, and 46 error messages for command context checking. For each command parsing and command checking error message in the utilities, an equivalent error message from the spec has been selected. Whenever the command parser or command checker signals an error, the number of the equivalent spec error is made available.

   There are no commands and no responses for which it is not possible to make an error. It is always possible, for example, to get the arguments wrong.

C. Because the spec is extremely vague about what error messages should be used in what situations, and in many cases several different choices seem reasonable, for the majority of errors signalled by the utilities, we do not know whether the spec error selected by the utilities is what is intended by the spec. In cases where the utilities signal an error in a command but the spec gives "Errors None", the utilities are clearly not in conformance with the letter of the spec. There are many of these, so we have not listed them all. For example, GetErrorInfo (Sec. 6.3.1.6) has no errors in the spec, but it is easy for the client to ask for information using an argument that is not the number of any defined error. This is clearly an error, so the command context checker signals an error.

8 Error response internal consistency

A. The spec does not prohibit sending an error response with a severity that does not match the severity assigned to an error of the given number or with text that does not match the text assigned to an error of the given number. (Sec. 8). Release 1.4 requires text matching but not severity matching.

B. The sender utility does not make either of these mistakes when sending error responses. The response parser detects both and signals an error if either occurs.

C. The utilities almost certainly do what the spec intends. Requiring the error response to be internally consistent was probably omitted from the spec inadvertently.

9 FindTool data response

A. In the description of the FindTool command, the spec says that data sent in response to a FindTool command should be "FoundTool". (Sec. 6.3.2.16). The example on page 63, however, does not show any returned data.

This is clarified in release 1.4 where it says no data response should be sent. (Sec. 6.3.2.16).

B. The server utility does not return a data response for FindTool.

C. The utilities seem to conform to the intent of the spec.

10 Get command arguments

A. The initial description in the spec of arguments to Get lists four items explicitly (X(), Y(), Z(), and Tool().A()) and allows "other methods that return axis information. Also temperatures and other dynamic properties of the system can be requested." (Sec 6.3.2.11).

The syntax of the fourth item is corrected to Tool.A() in release 1.4 (Sec 6.3.2.11). A fifth argument, R(), is allowed explicitly. This is the position of a rotary table. (Sec. 6.3.3.13). The spec does not explain anywhere what other dynamic properties are, or how to request information about them or temperature.

B. The rules in the utilities are:

- There must be one to seven arguments.
- Each argument must be R(), X(), Y(), Z(), Tool.A(), Tool.B() or Tool.C(), which may appear in any order.
- Each allowed argument must appear at most once.

The utilities do not allow temperature or other dynamic properties.

C. We believe what the utilities allows is what the spec intends with regard to axis information. The utilities do not conform to the spec with regard to temperature and other dynamic information (since we do not know what syntax should be used or what other dynamic properties are).

11 GetProp and GetPropE command arguments

A. The description in the spec of arguments allowed for GetProp and GetPropE gives only two explicitly allowed arguments, Tool.PtMeasPar.Speed() and Tool.GoToPar.Accel(), and then says "or other methods that return properties". It is not at all clear what is intended to be included. (Sec. 6.3.1.9 and Sec. 6.3.1.10).

In particular, it is not clear whether arguments such as Tool.PtMeasPar.Speed.Max() or Tool.GoToPar.Accel.Min() are intended to be allowed. Judging from the UML object model (Sec. 5.9) and the C++ code (Sec. A.6.6), the arguments Tool.PtMeasPar.MaxSpeed() and Tool.GoToPar.MinAccel() are intended to be allowed, even though that syntax is not described in Section 6. It seems likely that since the MinPROP syntax seems to be intended to be allowed, the PROP.Min syntax is not intended to be allowed. For several other properties not explicitly allowed, it is also unclear if they are intended to be allowed.

It is also not clear whether something like Tool.PtMeasPar(), whose value is not a primitive type is intended to be allowed.

It is also not clear whether CanChangeSpeed and CanChangeAccel are intended to be allowed. These are included in the UML model (Sec. 5.9) and the C++ code (Sec. A.6.5).

B. The utilities allow the MinPROP syntax (such as Tool.PtMeasPar.MaxSpeed() and not the PROP.Min syntax. The utilities do not allow an argument whose value is not a primitive type. The utilities do not allow CanChangeSpeed or CanChangeAccel.

   The rules in the utilities are:

   1. Each argument must have the pattern: key1.key2.key3()
   2. key1 must be FoundTool or Tool.
   3. key2 must be GoToPar or PtMeasPar.
   4. If key2 is GoToPar, key3 must be one of: Speed, MinSpeed, MaxSpeed, Accel, MinAccel, MaxAccel.
   5. If key2 is PtMeasPar, key3 must be one of: Speed, MinSpeed, MaxSpeed, Accel, MinAccel, MaxAccel, Approach, MinApproach, MaxApproach, Retract, MinRetract, MaxRetract, Search, MinSearch, MaxSearch.

C. We believe that all the arguments allowed by the utilities are intended to be allowed by the spec. The spec seems to intend that additional arguments be allowed.

12 GetXtdErrStatus and GetErrStatusE in case of error

A. The spec provides that in the case of any error, "The server will abort all pending transactions" and "The client must invoke the ClearAllErrors() method before the server can continue processing commands.". (Sec. 6.2.5). This is slightly inconsistent with "Errors with classification higher or equal 2 require ClearAllErrors.". (Sec 8.1). In release 1.4 an attempt to clarify this has been made (Sec. 6.2.5), but the language used is unclear and needs improvement.

   Currently, if an error of severity 2 or higher has been reported by the server, the spec requires that the server should respond to any command other than ClearAllErrors with an error message saying it is necessary to call ClearAllErrors in order to continue. In particular, this should be done if the client calls GetXtdErrStatus or GetErrStatusE. After ClearAllErrors executes without error, however, the status is OK and there are no extended error status details to report. This reduces the usefulness of GetXtdErrStatus and GetErrStatusE to almost nothing.

B. The utilities allow GetXtdErrStatus and GetErrStatusE commands to be executed while an error condition exists. It does not abort them if they are pending transactions.

C. We believe that, while what the utilities do does not conform to the spec as stated, this is problem with the spec and the utilities conform with what the spec intends.

13 GetXtdErrStatus data response

A. The spec does not require the server to send any data in response to a GetXtdErrStatus command, but says "The server may send one or more lines of status information ... as well as one or more Errors ...". (Sec 6.3.2.10).

B. The server utility does not send any data regardless of whether or not an error has occurred.

C. The utilities conform to the spec (since it makes no requirements), but the spec probably intends that one or more data responses be sent.

14 Home

A. The spec says "the server must be homed before the client can invoke methods that move the machine.". The spec says the Home command will home the machine but does not say whether anything else will cause the machine to "be homed". It is possible that the state of being homed can persist between sessions if the machine is not powered down. (Sec. 6.3.2.1).

B. After a StartSession command has been given, the utilities do not allow any commands (other than Home) that move the machine until the Home command has been given.

C. We are not sure what the spec intends.

15 OnMoveReportE command arguments

A. For OnMoveReportE, the spec explicitly allows Dis(<number>) and Time(<number>) arguments and references the section on Get regarding other arguments. (Sec. 6.3.2.7).

B. The rules in the utilities are:

   1. There must be zero to nine arguments.
   2. Each argument must be one of R(), X(), Y(), Z(), Tool.A(), Tool.B(), Tool.C(), Dis(<number>), or Time(<number>), which may appear in any order.

C. We believe that all the arguments allowed by the utilities are intended to be allowed by the spec. The spec seems to intend that additional arguments be allowed.

16 OnPtMeasReport command arguments

A. The spec is not clear about what arguments are allowed in OnPtMeasReport. In particular, it does not list IJK (or any other parameters) explicitly or (by its reference to parameters allowed with Get) imply implicitly that IJK should be allowed. (Sec. 6.3.2.6 and Sec. 6.3.2.11). In release 1.4, IJK is explicitly allowed as a parameter in OnPtMeasReport. (Sec. 6.3.2.6).

B. The utilities do not allow IJK as a parameter in OnPtMeasReport. This will be changed in future versions of the utilities.

C. The utilities currently do not do what was apparently intended by the spec.

17 OnScanReport with no arguments not allowed

A. The spec says nothing about whether an OnScanReport command with no arguments is allowed. If such a command is allowed, it is not clear what the server should do after receiving such a command. (Sec. 11.1.2). Having no parameters is not allowed for OnPtMeasReport (Sec. 6.3.2.6)

B. Because of the confusion caused by allowing no arguments and because it seems preferable to have OnScanReport be consistent with OnPtMeasReport, the utilities do not allow OnScanReport with no arguments. An error is signalled if such a command is received.

C. We are inclined to think the spec writers intended not to allow no arguments to OnScanReport, but did not write it down. Thus, we think the utilities conform with the intent of the spec.

18 Parameter out of range error message

A. The spec defines the error message 0504 "Parameter out of range" but does use it anywhere. (Sec. 8.2). In release 1.4, this error is required to be sent if a SetProp command is given containing a value that is out of range. (Sec. 6.3.1.11).

B. The utilities do what is required by release 1.4, as just described.

C. This is probably what was intended in the spec.

19 Preserving state between sessions

A. In the documentation of EndSession the spec says "The following states of the server are preserved upon connection of the next client: Active tool, Active coordinate system". (Sec. 6.3.1.2). The spec does not say what the states of a server are or what states (if any) should persist across a power down. Release 1.4 specifies that on a start session, OnPtMeasReport items are set to X(), Y(), and Z() and OnScanReport items are set to to X(), Y(), Z(), and Q(). (Sec. 6.3.1.1).

B. In the server utility, no states persist across exiting the utility. The following persist between an EndSession and a StartSession as long as the server utility is not shut down:

   - active tool,
   - active coordinate system,
   - axis positions,
   - the entire tool_changer, including all tools and their parameters.

   The following are reset when an EndSession is executed:

   - OnMoveReport items are set to report nothing,
   - OnPtMeasReport items are set to report nothing,
   - OnScanReport items are set to report nothing,
   - being homed is set to not homed.

   StartSession resets no OnReport items.

C. The utilities conform to the spec regarding the active tool and the active coordinate system. In view of what release 1.4 does, the utilities do not seem to conform to the intent of the spec regarding OnReport items.

20 PtMeas command arguments

A. The spec does not say whether PtMeas is allowed to have none of X(), Y(), or Z() as arguments. (Sec. 6.3.2.13). Release 1.4 requires at least one of them to be used. (Sec. 6.3.2.13).

B. The utilities require that at least one of X(), Y(), or Z() be used as an argument.

C. The utilities seem to do what the spec intends.

21 PtMeas target point out of work volume

A. The spec does not say whether having the target point of a PtMeas command out of the work volume is an error. It does say that having the end-of-search point out of the work volume is not an error. (Sec. 6.3.2.13).

B. The command context checker (and hence the server utility) signals an error if the target point is out of the work volume.

C. We do not know if what the utilities do is what is intended by the spec.

22 ScanInCylEndIsPlane command arguments

A. The spec does not place any requirements on the number of through stop plane (n) or on the average distance between adjacent points (StepW). (Sec. 11.3.6).

B. The utilities require n to be a positive integer and require StepW to be positive.

C. Since n does not make sense as anything but a positive integer, and StepW is a distance, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW or n to be negative with the understanding that if a value is negative its absolute value should be used.

23 ScanInCylEndIsSphere command arguments

A. The spec does not place any requirements on the diameter of the sphere (Dia) or on the average distance between adjacent points (StepW). (Sec. 11.3.5).

B. The utilities require Dia and StepW to be positive.

C. Since Dia and StepW are distances, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW or Dia to be negative with the understanding that if a value is negative its absolute value should be used.

24 ScanInPlaneEndIsCyl command arguments

A. The spec does not place any requirements on the number of through the cylinder (n), on the average distance between adjacent points (StepW), or on the diameter of the cylinder (d). (Sec. 11.3.4).

B. The utilities require n to be a positive integer and require StepW and d to be positive.

C. Since n does not make sense as anything but a positive integer, and d and StepW are distances, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW, d, or n to be negative with the understanding that if a value is negative its absolute value should be used.

25 ScanInPlaneEndIsPlane command arguments

A. The spec does not place any requirements on the number of through the plane (n) or on the average distance between adjacent points (StepW). (Sec. 11.3.3).

B. The utilities require n to be a positive integer and require StepW to be positive.

C. Since n does not make sense as anything but a positive integer, and StepW is a distance, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW or n to be negative with the understanding that if a value is negative its absolute value should be used.

26 ScanInPlaneEndIsSphere command arguments

A. The spec does not place any requirements on the diameter of the sphere (Dia) or on the average distance between adjacent points (StepW). (Sec. 11.3.2).

B. The utilities require Dia and StepW to be positive.

C. Since Dia and StepW are distances, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW or Dia to be negative with the understanding that if a value is negative its absolute value should be used.

27 ScanOnCircle command arguments

A. The spec does not specify what the sign of the total angle of scan (delta) must be. It makes sense to allow delta to be either positive or negative to allow for either clockwise or counterclockwise motion. So, presumably, delta can be positive or negative. The spec is not explicit about what direction is intended by a positive angle and what direction is intended by a negative angle. The figure shows a counterclockwise angle but does not specify whether delta is positive or negative in this case. (Sec. 11.2.2). In release 1.4, it is specified that delta may be positive or negative and is positive counterclockwise.

The spec also does not specify what sign the incremental angle (StepW) should be. It is confusing if, for example, alpha is positive and StepW is negative. Does this mean to go in the opposite direction from alpha until reaching the point indicated by alpha? Or is the sign of StepW supposed to be irrelevant and the scan should always proceed in the direction of alpha? It would be reasonable to require either (1) that StepW always have the same sign as alpha, or (2) that StepW must always be positive with the understanding that the step is always taken in the direction of alpha.

B. The utilities allow delta to be positive or negative.

The utilities take a positive delta to mean a counterclockwise angle as viewed from the positive Z-axis.

The utilities require StepW to be positive. The step is always taken in the direction of alpha.

C. The utilities seem to conform to the intent of the spec regarding delta. We do not know what the spec intends regarding StepW.

28 ScanOnCircleHint command arguments

A. The spec does not place any requirements on the Displacement or Form arguments. (Sec. 11.2.1).

B. The utilities require the Displacement and Form arguments to be non-negative.

C. Since displacement is a distance, and form deviation is calculated as a non-negative number, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow Displacement or Form to be negative with the understanding that if a value is negative its absolute value should be used.

29 ScanOnLine command arguments

A. The spec does not place any requirements on the average distance between adjacent points (StepW). (Sec. 11.3.4).

B. The utilities require StepW to be positive.

C. Since StepW is a distance, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow StepW to be negative with the understanding that if it is negative its absolute value should be used.

30 ScanOnLineHint command arguments

A. The spec does not place any requirements on the Angle or Form arguments. (Sec. 11.2.3).

B. The utilities require the Angle and Form arguments to be non-negative.

C. Since angle and form deviation are calculated as non-negative numbers, what the utilities do probably matches the intent of the spec. It is possible that the spec intends to allow Angle or Form to be negative with the understanding that if a value is negative its absolute value should be used.

31 SetProp command arguments

A. The description in the spec of arguments to SetProp lists two items explicitly (Tool.PtMeasPar.Speed(100) and Tool.GoToPar.Accel(10)) and allows "other methods that set properties". (Sec. 6.3.1.11).

The SetProp section of the spec does not say whether arguments such as Tool.PtMeasPar.Speed.Max(1000) or Tool.PtMeasPar.MinSpeed(1000) are allowed, but resetting maxima and minima is later explicitly prohibited. (Sec. 6.3.6 and Sec. 6.3.7).

B. The rules in the utilities are:

1. There must be at least one and not more than MAXPROP arguments.
2. Each argument must have the pattern: key1.key2.key3(<number>)
3. key1 must be FoundTool or Tool.
4. key2 must be GoToPar or PtMeasPar.
5. If key2 is GoToPar, key3 must be either Speed or Accel.
6. If key2 is PtMeasPar, key3 must be one of: Speed, Accel, Approach, Retract, or Search.

C. We believe that all the arguments allowed by the utilities are intended to be allowed by the spec. The spec seems to intend that additional arguments be allowed.

32 SetProp data response

A. Under the Data heading for SetProp, the spec says "The format is defined by the method enumerated". (Sec. 6.3.1.11). This seems to imply that data should be returned. However, it is not clear what data should be returned. Should the returned value be the value in the command or the value actually set (these will differ if the commanded value out of range)? Also, the example on page 63 does not show any returned data at all, so perhaps no data is supposed to be returned, and the quoted sentence above is there only as a result of a cut-and-paste that was supposed to be edited but was not.

B. The server utility returns no data for SetProp.

C. We have no idea whether the utilities do what the spec intends.

33 SetProp when parameter out of range

A. The spec says if an attempt is made to set a parameter, and the requested value is outside the range given by the maximum and minimum values for the parameter being set, the value should be set to either the maximum or the minimum. (Sec. 6.3.6 and Sec. 6.3.7)

B. In the utilities, if a SetProp command is received and the value in the command is greater than the maximum allowed for the parameter being set, then the value is set to the maximum allowed. Also, if the value in the command is less than the minimum allowed, then the value is set to the minimum allowed.

C. We believe this is what the spec intends, although the spec is not explicit.

34 StartSession and errors

A. If, after EndSession is executed, a command is sent by the client that causes an error, the spec requires that ClearAllErrors be sent to clear the error before it is possible to execute StartSession. (Sec. 8). But the spec does not provide that any command other than StartSession can be executed when no session is in progress, so it seems implicit (although it is not explicit!) that ClearAllErrors cannot be executed when no session is in progress. Thus, it is not possible to restart the server in this situation.

   Release 1.4 fixes some of this, but fails to state explicitly that StartSession can be called when no session is in progress even if the server is in an error state.

B. The utilities allow StartSession to be called any time no session is in progress, regardless of whether the server is in an error state or not. StartSession called when the server is in an error state has the effect of ClearAllErrors followed by StartSession when there server is not in an error state.

C. We believe the utilities conform to the intent of release 1.4 of the spec.

35 Tag type matching command type

A. The spec does not explicitly prohibit sending an event command with a non-event tag or sending a non-event command with an event tag. (Sec. 6.2.1).

B. The utilities prohibit both.

C. The utilities almost certainly do what the spec intends. The prohibition was probably omitted from the spec inadvertently.